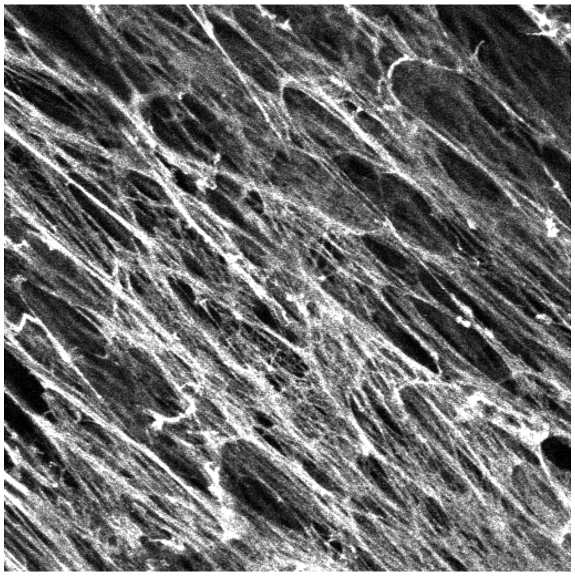


CLASSIFICATION OF EXTRACELLULAR MATRIX

Using Curvelet based Feature Extraction and Deep-learning

Avrajit Ghosh
Summer Research Intern at Inria Sophia Antipolis Mediterranean

Under Supervision of :
Laure Blanc-Feraud (Director of Research at CNRS)
Xavier Descombes (Director of Research at Inria)

Contents

1 Abstract

ECM (Extracellular matrix) is a three dimensional network of extracellular macromolecule such as colagen, glycoprotien and enzymes that provide structural and biochemical support to the surrounding cells. Depending on the molecular composition of the ECM, it is classified into four variants namely-A+,A-B-,B+ and A+B+. In this report we examine the topography of the various ECM variants based on their images acquired by confocal microscope and coverslip scanner. Discrete Curvelet Transform has been used as a tool to extract the information about the scale and orientation of the fibre molecules. Based on the extracted curvelet coefficients from the images, classification is performed to distinguish the ECM variants. Finally a comparision has been drawn between the images acquired by that of the confocal microscope and coverslip scanner. As classification based on feature extraction by curvelets only gave an accuracy of 69.28%, we decided to apply a deep learning technique known as Convolutional Neural Net to see if we can improve the accuracy.A pretrained Convolutional Neural Net (GoogleNet) is thus used for Transfer Learning and necessary layers has been changed to adapt the net to train and classify Extracellular matrix images.

2 Introduction

The internship is a blend of signal processing and machine learning. I chose to do an internship at I3S lab with team Morpheme because working with researchers on cutting-edge signal processing projects would prepare me to be an accomplished researcher in this field in my future.
The aim of this internship is to classify images of extracellular (ECM) matrices. The ECM is a complex network of macromolecules that is secreted by cells. The ECM is in close contact with the cells.

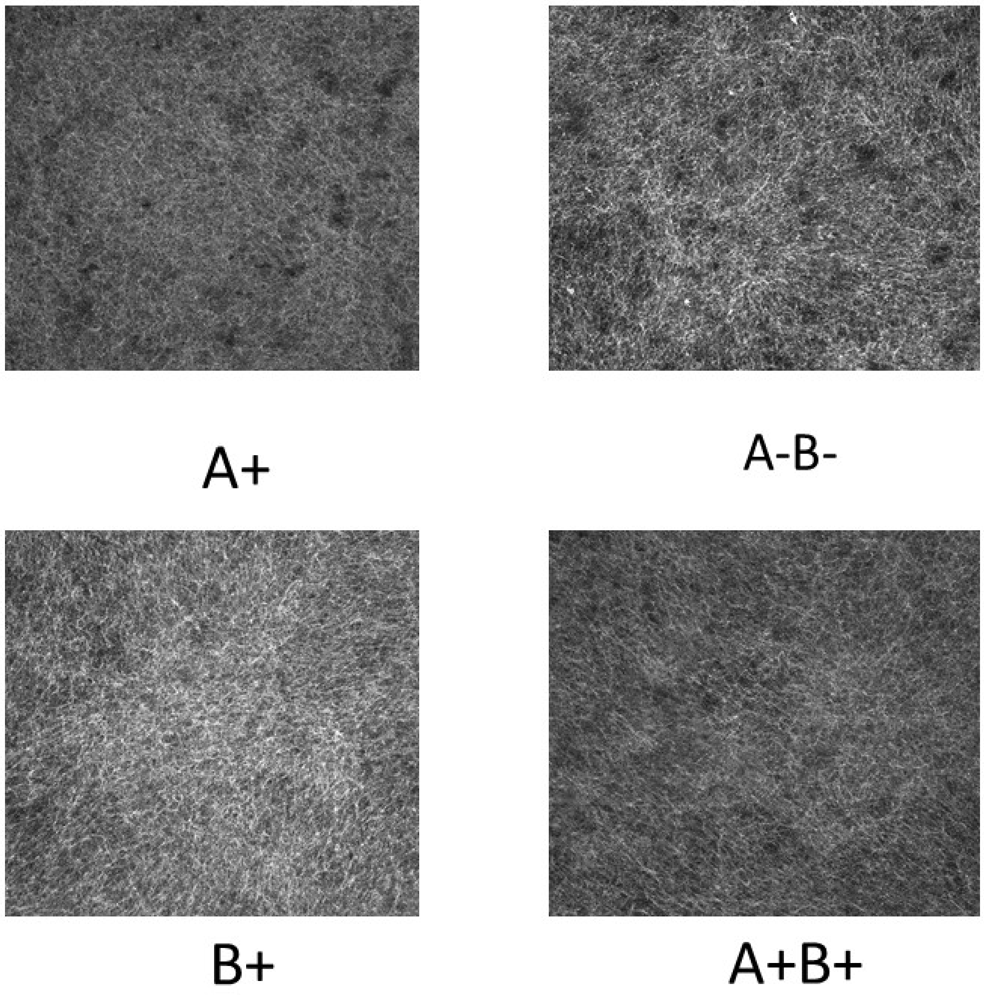


Figure 1: Variants of Extracellular Matrix

In this project we find ways to classify any ECM image into any one of the four variants- A+, A-B-,B+,A+B+. It is hard to tell them apart with the help of our bare eyes but each ECM variant has a unique curve orientation and thickness level. That's why we have used the Curvelet Transform to extract features from the image. We would like to obtain good classification results with the Curvelet

transform because the way we use it has a real physical meaning. In our second approach we have used Transfer learning using a pretrained convolutional Neural Net known as GoogleNet. These two approaches have been applied on the images acquired by two high resolution optic device-
1) Confocal Microscope 2) Coverslip Scanner
The first part of this report is a description of the general framework of my internship. The second part is devoted to the description of the extracellular matrix. Then we introduce the classification problem using curvelet feature extraction and deep learning. Finally we present the results we obtain and draw a comparision using both the classification methods and acquisition technique.

3 Presentation of my team

I did my internship in the MORPHEME team, based in Sophia-Antipolis and which is a joint project between three institutes: Inria, CNRS and the University Nice-Sophia Antipolis. The people in this team come from three laboratories: Inria Sophia Antipolis-Méditeranée, the Computer Science, Signals and Systems Laboratory (I3S) and the Institute of Biology Valrose.

3.1 Inria Sophia Antipolis- Mediterranean

The Inria is the french institute for computer science and applied mathematics. It was
founded on the 3rd January 1967. Its main focus is to promote scientific excellence for tech- nology transfer and society. Today, it employs 2,700 people. Research is organized in "project
teams" (such as MORPHEME) that brings people from different fields and with complementary skills to focus on specific scientific projects. According to the institution, this model enables the Inria to explore new approaches to provide an efficient response to the multidisciplinary and application challenges of the digital transformation. Inria has eight research centers (in Bordeaux, Grenoble, Lille, Nancy, Paris-Rocquencourt, Rennes, Saclay, and Sophia Antipolis) and also contributes to academic research teams outside of those centers.

Inria established itself on the Sophia Antipolis site in 1981. At that time it was the second Inria research center created in France after the one in Paris. In 2008, the Inria extended its activities to Montpellier and Bologna (Italy) and changed its name from "Sophia Antipolis" to "Sophia Antipolis – Méditerranée". Today the centre comprises:
1)600 people including 400 scientists.
2)working with 35 teams: 29 are based in Sophia-Antipolis, five in Montpellier and one in Bologna. Most of these teams are joint teams between one or more partners.
3)people from 50 nationalities are represented.

3.2 The Computer Science, Signals and Systems Laboratory (I3S)

The I3S laboratory is one the most important research facility for information technology on the Côte d'azur and one the first that settled on the Sophia-Antipolis site. Today, 300 people are employed by the laboratory (including phD students), most of the permanent members are professors at the University of Nice-Sophia Antipolis. There are four main axes of research, one of them is called Signal Image and Systems (SIS) and the MORPHEME team is part of it.

3.3 The Institute of Biology Valrose

The Institute of Biology Valrose (iBV) in Nice is an international research centre funded by CNRS, INSERM, the University of Nice-Sophia Antipolis, the Regional Cancer Research Center and the European Union. The focus of the Institute is to understand the basic principles
governing the development of normal cells, tissues and embryos and those leading to pathogen- esis and cancer. It brings together research teams with complementary areas of expertise and
with a common interest in translating basic research into knowledge for the clinic.
There are 25 different teams working at the iBV. However for my internship, we only work with one: the team of Ellen Van Obberghen-Schilling which focus is on the extracellular matrix (ECM) especially the role it plays in cancer development. A description of what the ECM is and what functions it has is done in later part of this report.

3.4 The Morpeheme team

The scientific objectives of MORPHEME are to characterize and model the development and the morphological properties of biological structures from the cell to the supra-cellular scale. Being at the interface between computational science and biology, the focus is made on understanding the morphological changes that occur during development combining in vivo imaging, image processing and computational modeling.

4 The extracellular matrix

4.1 Primary function of ECM

In biology, the extracellular matrix (ECM) is an agglomeration of extracellular molecules that are secreted by cells that provides structural and biochemical support to the surrounding cells. The composition of the ECM depends on the cells that are lying around it, however its main functions remain cell adhesion and cell-to-cell communication.
One has to understand that the ECM nanotopography is a regulator of cell functions. Indeed the ECM affects cells behavior by supplying spatial and mechanical cues that cells respond to. Using physical interaction as well as soluble chemical factors, the ECM also regulates cell processes (such as differentiation for example)

4.2 Components of ECM

The ECM is formed by the association of three different types of molecules:
1) fibers: collagen and elastin
2) proteoglycans such as fibronectin
3) polysaccharides In this project, the first two are more important for us because they are the molecules responsible for the filaments.

The biology laboratory, the Institut de biologie de Valrose, we are working with is also interested in trying to characterize the ECM topography between healthy patients. As a matter of fact, it seems that there are four possible different chemical compositions (in this case this is how the fibronectin is synthesized within the cell that leads to four distinct fibronectin molecules), they are known as variants. The difference of topography between those variants has never been proven but if we could characterize them it would be a great step in this direction. In the tests featured later in this report, the classification was only done for the four variants, we did not focus on the ECM of ill patients. Nevertheless, our work can be applied to patients with cancer as well.

5 Image classification using Feature Extraction by Curvelets

Before going into the property of the curvelets, let us introduce the classification pipeline that has been used here. It consists of three steps.
1)Input Database- The input of the algorithm is a set of 280 images belonging to 4 classes. It is further split into the training set and the test set.
2) Learning- We used the training set to learn what every one of the classes look like, in other words we extract numerical features which characterize each class. This step is referred to as training the classifier or learning the model.
3) Testing- Finally we need to evaluate the quality of the classifier by asking it to predict labels for images it has never seen before. We will then compare the true labels of these images to the ones predicted by the classifier. We are hoping that a lot of the predictions match up with the true answers

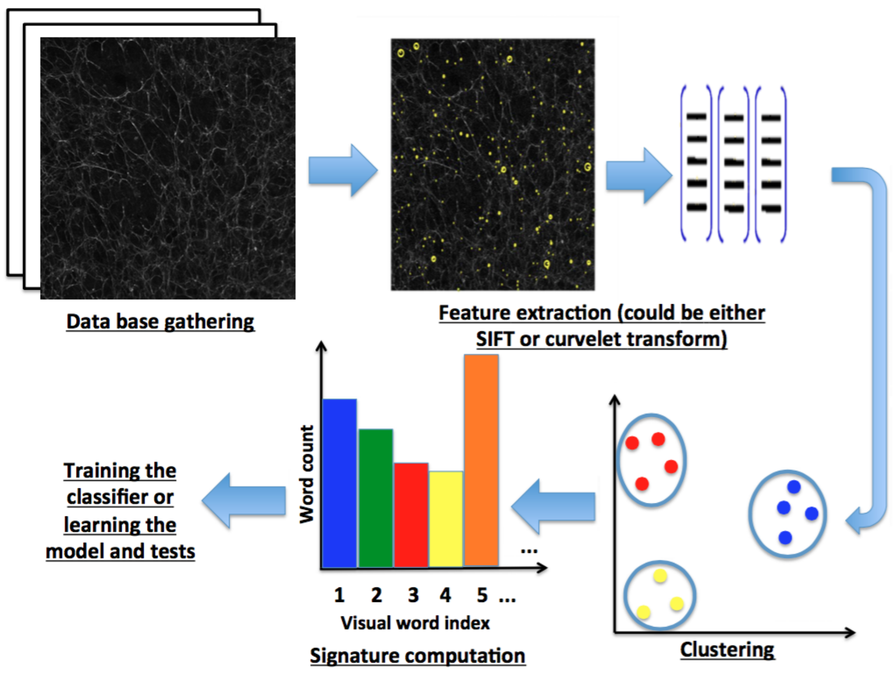

Data base gathering
Feature extraction (could be either SIFT or curvelet transform)
Training the classifier or learning the model and tests
Word count
1 2 3 4 5 ...
...
Visual word index
Signature computation
Clustering

Figure 2: Classification pipeline

The learning part includes:

1) Feature extraction by obtaining the curvelet coefficients of the images by Discrete Curvelet Transform
2) Keypoint Extraction by k-means clustering
3)Obtaining the image signature of each image from the clustered set of data
4) Train DAG-SVM classifier using the obtained image signature

5.1 The Curvelet Transform

To consider the Curvelet transform we first introduce the discrete wavelet transform (DWT). The latter has been applied quite a lot in the signal processing field but it has some limits. In fact, the 2D dyadic DWT has only three possible orientations (horizontal, vertical, diagonal), so it is not suited for characterizing textures or edges (for example) in an image. To overcome these limits, Candès and Donoho proposed a new multidirectional and multiscale transform - the Curvelet transform - adapted to the detection of curves and contours. The basis function are long, more or less fine structures.

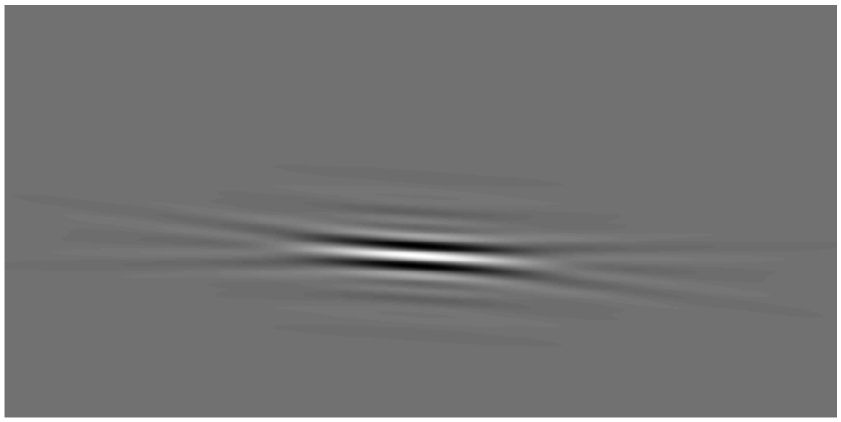


Figure 3: Picture of a curvelet at scale 6 and orientation $-\pi/64$

Curvelets occur at all scales, locations and orientations. The continuous-time Curvelet transform is constructed in ^{2}. We take x as a spatial variable, ω a frequency domain variable with r and θ the polar coordinates in the frequency domain. We then give ourselves two windows, $W(r)$ the radial window and $V(t)$ the angular one. Both of them are smooth, non-negative and real valued. W takes, as input, positive real arguments and is supported on $r \in [1/2; 2]$ whereas V takes real arguments and is supported on $t \in [-1; 1]$. We can use any window we want for W and V but they have to fulfill the admissibility conditions:

$$\sum_{j=-\infty}^{+\infty} W^2(2^j r) = 1,\, r \in [3/4; 3/2] \tag{1}$$

$$\sum_{l=-\infty}^{+\infty} V(t-l) = 1,\, t \in [-1/2; 1/2] \tag{2}$$

Now, for each scale j, $j \geq j_0$, we note U_j the frequency window defined in the Fourier domain by:

$$U_j(r,\theta) = 2^{-3j/4} W(2^{-j}r) V(\frac{2^{\lfloor j/2 \rfloor}\theta}{2\pi}) \tag{3}$$

where $\lfloor j/2 \rfloor$ is the integer part of $j/2$. We can now define the waveform $\varphi_j(x)$ using its Fourier transform $\hat{\varphi}_j(\omega) = U_j(\omega)$. φ_j can be seen as the "mother" curvelet since, at scale j, all curvelets are obtained by rotations and translations of φ_j.
The rotation parameter θ_l is designed such that the spacing between consecutive angles is scale dependent:

$$\theta_l = 2\pi 2^{\lfloor j/2 \rfloor} l, \text{ where } l = 0, 1, \dots \text{ such that } 0 \leq \theta_l \leq 2\pi$$

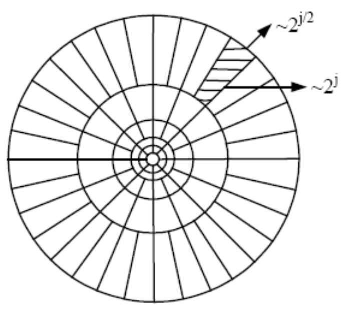
The translation parameter is $k = (k_1, k_2) \in^2$.

The curvelet is now defined as a function of x, at scale 2^{-j}, orientation θ_l and position $x_k^{(j,l)} = R_{\theta_l}^{-1}(k_1 2^{-j}, k_2 2^{-j/2})$ by:

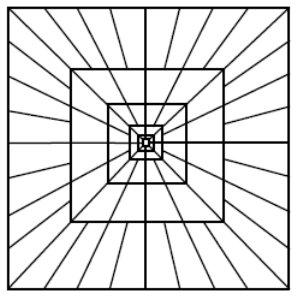
$$\varphi_{j,l,k}(x) = \varphi_j\left(R_{\theta_l}(x - x_k^{(j,l)})\right) \tag{4}$$

where R_θ is the rotation matrix by θ radians and R_θ^{-1} its inverse.
A curvelet coefficient is then simply the inner-product between an element $f \in L^2(^2)$ and a curvelet $\varphi_{j,l,k}$,

$$c(j,l,k) = f, \varphi_{j,l,k} = \int_2 f(x)\overline{\varphi_{j,l,k}(x)}dx \tag{5}$$



(a) Continuous Curvelet tiling



(b) Discrete Curvelet tiling

Figure 4: Illustration of the curvelet tiling of frequency [3]

Candes proposed two different implementations of the discrete Curvelet transform, via Unequispaced Fast Fourier Transforms (USFFT) or via Wrapping. In our project we decided to use the second implementation because at the finest scale we can choose to have a wavelet transform or a curvelet transform. This choice is not allowed by the other implementation.

The software we worked with is Curvlab and can be found at `Curvelet.org`. Generally the output of the curvelet transform is a big vector such that each entry corresponds to a specific scale. If we look in one of the entries, there are matrices containing the curvelets coefficients for a specific orientation. If we lay them end to end, there is quite a large number of them (it depends on the size of the image but for picture of 1024×1024, you can get more than seven millions coefficients). As the reader may already know, the coefficients that come from the coarsest scale are low frequency coefficients that are not typical of the image structure. Please note that in the further described proceedings, the low frequency coefficients are considered already removed.

In an attempt to downsize this amount of coefficients, we decided to compute the energy of the curvelet transform as the sum of all the coefficients squared. Then it is up to the user of the progam to choose which amount of this energy he wants to keep, for example if you decide to keep 80% of the energy for an image 1024×1024 you can go from more than 7 millions coefficients to a little bit less than six thousands. This is done by only keeping the largest curvelets coefficients no matter at which scale (but the coarsest) and which orientation they have been detected.

Finally, a keypoint is a three dimensional vector composed of a scale an orientation and the curvelet coefficients.

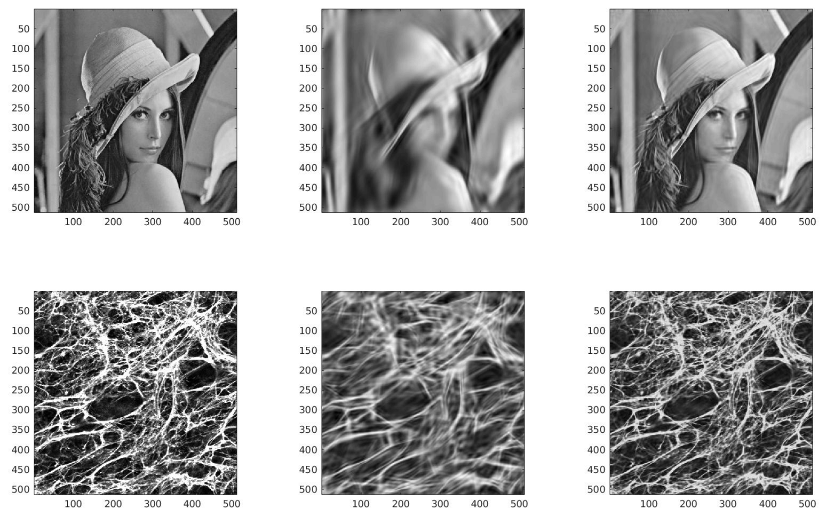


Figure 5: Partial reconstruction of images using 2 levels of energy (left: Original image, center: 50% of energy kept, right: 90% of energy kept)

By designing the keypoints in this way, we loose the spatial information. In an attempt to keep for future proceedings or for another application, we decide to represent the curvelets involved in our decomposition of the image. The results can be seen in the following figure. This representation is for a specific scale and the different colors stand for the number we have assigned to particular curvelet, purple means that there are no curvelets kept at all.

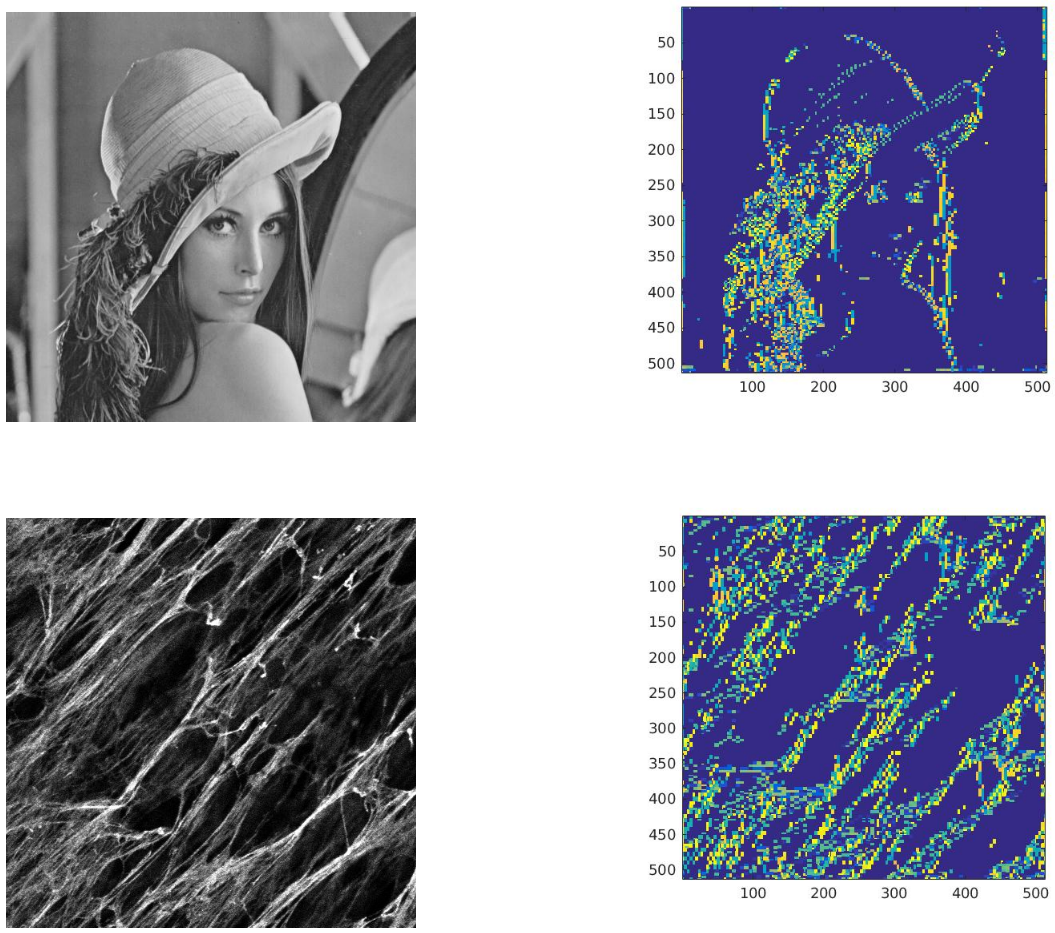


Figure 6: Illustration of the curvelets kept for a given level of energy at a particular scale (Left: original image, right: Number of the curvelets)

In many scientific and engineering fields, such as medical imaging for example, the data is inherently three dimensional. Since properties such as scale, position, orientation, scaling can be extended to 3D and with the huge computational power we have nowadays, 3D transforms (like the 3D curvelet transform) can be a great tool to process and analyse large dataset.

5.2 Invariance to rotation

Orientation of the fibers in the ECM image is a significant factor for the classification algorithm. However during the acquisition process, the samples of ECM may be differentially oriented which may effect the classification accuracy. Hence to ensure that they follow the same privileged direction, this step is performed.
First we calculated the dominant orientation of each image by calculating it's gradient vector and then rotated the image by an angle known as Dominant Orientation. For an image f ϵ $\mathbb{R}^2$, the gradient vector is $G = (g_x, g_y)$ with it's magnitude $|G| = \sqrt{g_x^2 + g_y^2}$ and orientation $\theta = arctan(\frac{g_y}{g_x})$. Then the

dominant orientation α is calculated as

$$\alpha = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}|G_{ij}|^2.\theta ij}{\sum_{i=1}^{N}\sum_{j=1}^{N}|G_{ij}|^2}$$

(6)

After orintation the dominant orientation of all the images is towards the horizaontal direction.

5.3 Compatibility of Extracellular Matrix in Curvelet basis

An image in spatial domain contains a lot of information so it is tough to classify an image in the spatial domain because of the time and computational complexity. So it is preferred to transform an image into a sparse domain so that we can get a very good approximation of the same image using very less amount of coefficients. The conventional transform domains includes the Fourier and wavelet domains. The basis function of the Fourier domain is a complex exponential whereas in wavelet domain it is a mother wavelet corresponding to a specific shape and oscillation. Let f be the original image and f(m) be the m-term approximation of the same image in a sparse domain. If f is an image containing curves and edges and $f(m)_{Fourier}$ is the m-term approximation of image f in Fourier domain,then -

$$||f - f(m)_{Fourier}|| \leq Cm^{-0.5} \tag{7}$$

whereas if $f(m)_{Wavelet}$ is the m-term approximation of image in Wavelet domain, then-

$$||f - f(m)_{Wavelet}|| \leq Cm^{-1} \tag{8}$$

Curvelet transform is a new multidirectional and multiscale transform proposed by Candes and Donoho for the detection of curves and contours. The basis function in curvelets are fine long fine structures having specific thickness (referred to as scale), a fixed orientation and fixed coordinates corresponding centre of the curve in the special domain. Curvelet domain provides the smallest asymptotic error for fixed number of terms [12] .

$$||f - f(m)_{Curvelet}|| \leq C.(log(m))^3 m^{-2} \tag{9}$$

Moreover it is also evident by looking at an ECM image that curvelet provides the best representation as a single curvelet basis looks very similar to an ECM molecular fibre. Curvelet coefficient c(j,l,k) is obtained by taking the inner product of f with $\psi(j,l,k)$, the basis function for the curvelet with scale parameter j, orientation parameter l and k=(k1,k2), the 2d coordinates of the centre of the spatial domain of the image.

$$c(j,l,k) =< f, \psi(j,l,k) > \tag{10}$$

As a very good approximate of the image is obtained by using a very few curvelet coefficients, we took only those largest curvelet coefficients that make up 85% of the total curvelet coeffiicents. For example a 512*512 image would result in more than 4 million curvelet coefficients but preserving only the 60000 largest coefficients would result in 85% retainment of the original image which is a huge improvement in terms of image classification. We are interested in the composition of curvelets belonging to a specific scale and orientation in ECM image so the three important parameters extracted are j(scale), l(orientation) and $|c(j,l)|^2$. It is to noted that the spatial information k of the curvelets is ignored.

5.4 Generation of image signature

We are using curvelet coefficients that compose 85% of the total energy of the image. Each curvelet coefficient is represented as 3*1 vector (j,l,$|c(j,l)|$). However each image has almost 60000 such coefficients which is our bag of features, thus it is not practical to perform classification using so much data. So we generate some specific keypoints using K-means. First we calculate the 3*1 curvelet coefficients for all images in our dataset (280 images). Then we divide the curvelet coefficient space (3-d space) into k clusters, each cluster is represented by the centroid of the cluster.given a set of n observations (x1; x2; ...; xn), the k-means clustering algorithm ($k \leq n$) will partition the data into k clusters (S1; S2; ...; Sk) in order to find

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{11}$$

It is to noted that k is to be carefully chosen as increasing k would minimise the SSE and provides a better approximation of the image but at the expense of the increased computational time. Experiments have been performed by noting the SSE for various values of k and selecting k using elbow method.

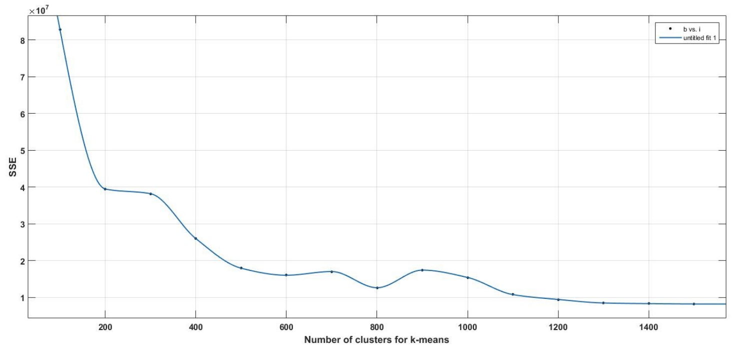


Figure 7: Variation of SSE with number of centroids

Let N(i) be the number of curvelet coefficents belonging to cluster Si , then the image signature is a vector

$$Q(i) = N(i) / \sum_{i=1}^{k} N(i) \tag{12}$$

for i=1,2,3...k or more precisely each element in the image signature represents proportion of the coefficents out of total coefficents belonging to a specific cluster.

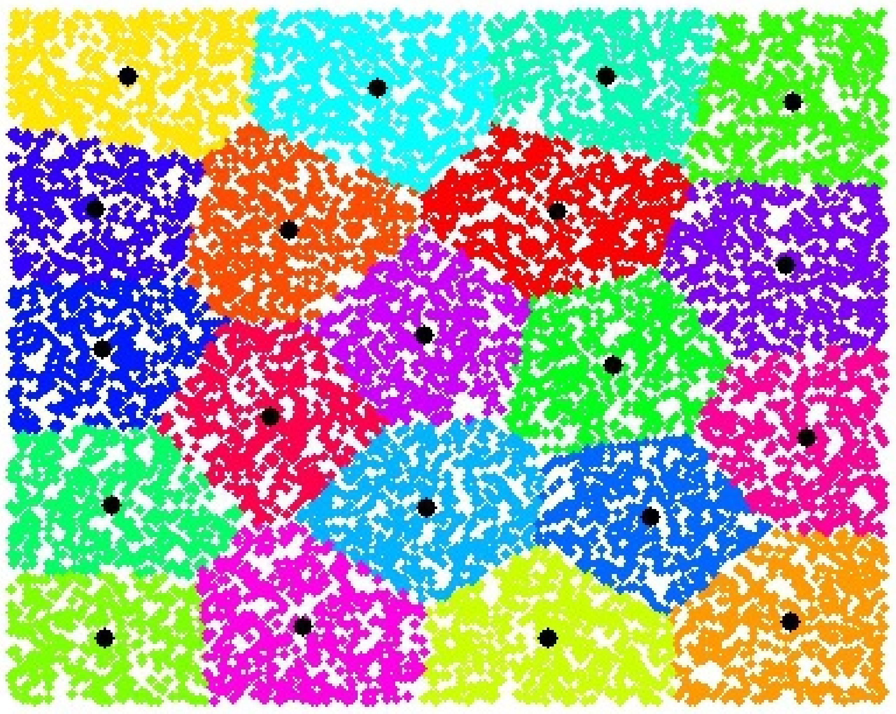


Figure 8: K-means clustering on 5000 data points with k = 20

5.5 Classification using DAG-SVM

SVM (Support Vector Machine) is a supervised machine learning model in which a hyperplane is constructed (called as margin) that isolates two groups of points in N-dimension such that distance from margin to the group of points is the largest.
Mathematically, we have a training set $\{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), ..., (\overrightarrow{x_n}, y_n)\}$ where each $\overrightarrow{x_i}$ is a p-dimensional vector and y_i is just a label, 1 or -1, depending on the class of the point. We want to find an hyperplane that separates the points with label -1 from the points with label 1. The equation of such plane can be written as:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = 0 \tag{13}$$

where $\overrightarrow{w}$ is a normal vector to the hyperplane. Then there is two ways of finding this hyperplane:

- If the data are linearly separable, we can find two other hyperplanes with equations $\overrightarrow{w} \cdot \overrightarrow{x} - b = 1$ and $\overrightarrow{w} \cdot \overrightarrow{x} - b = -1$ such that the first hyperplane we were looking for lies halfway between them. The margin, the distance between the two hyperplanes described above, is $\frac{2}{\|\overrightarrow{w}\|}$. So to maximize the margin we want to minimize $\|\overrightarrow{w}\|$. This is the hard-margin. In order to prevent the points from falling into the margin a constraint is added for each point:

$$y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - b) \geq 1 \, \text{for all } i \in [1; n] \tag{14}$$

  If we put all the pieces together we find the following optimization problem:

$$\text{Minimize} \, \|\overrightarrow{w}\| \, \text{subject to } y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - b) \geq 1 \, \text{for all } i \in [1; n] \tag{15}$$

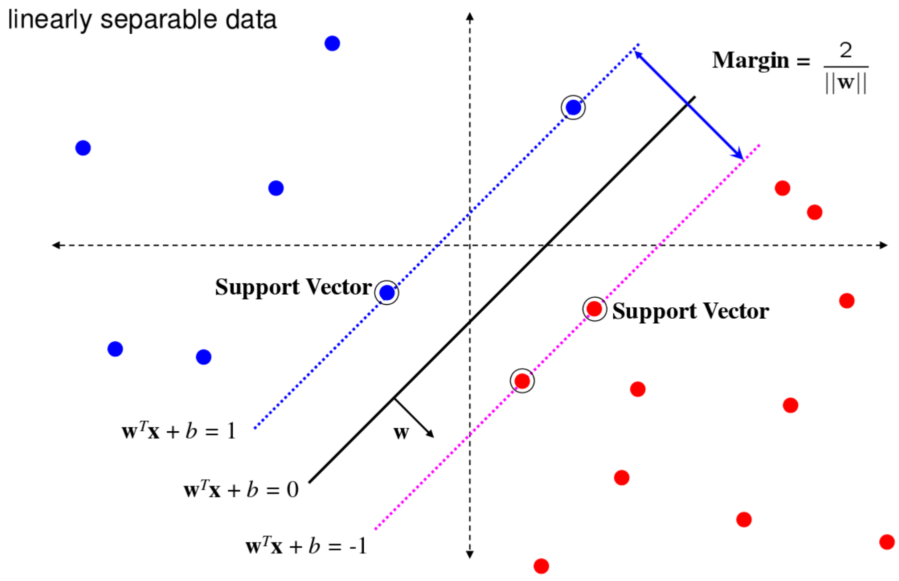

linearly separable data
Margin = $\frac{2}{\|\mathbf{w}\|}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Figure 9: Support Vector Machine

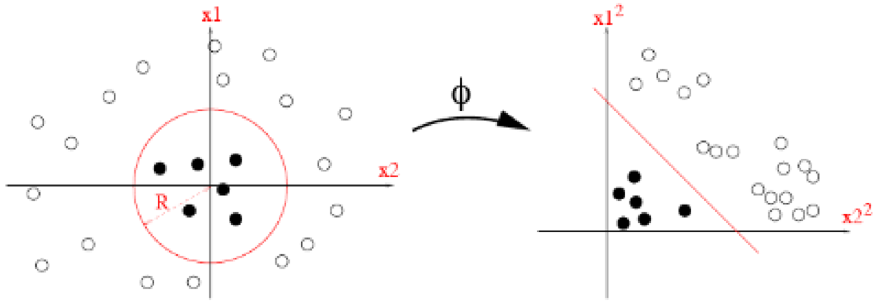


Figure 10: Kernel Trick

Apparently $\overrightarrow{w}$ of the margin is chosen such that there is no error on the training data. But in general we prefer a trade off between the margin and the number of mistakes on the training data. This is done by introducing a slack variable ξ_i such that for each each point i, if $0 \leq \xi_i \leq 1$, then the point is between margin and correct side of hyperplane and if $\xi_i \geq 1$, then the point is misclassified. The optimisation problem now becomes (where C is the regularization parameter) such that-

$$\underset{w,b,\xi}{\text{Minimize}}\; C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|\overrightarrow{w}\|^2$$

$$\text{subject to}\, y_i(\overrightarrow{w}\cdot\overrightarrow{x_i} - b) \geq 1 - \xi_i,\ \text{and}\, \xi_i \geq 0,\ \forall i \in [1;n]$$

Restrictions on the margin can be specified by chosing a proper value of C:

1) small C allows constraints to be easily ignored i.e it can misclassify few points but provides a large margin.
2) Large C makes constraints hard to ignore and provides a narrow margin.
3) C tends to infinity means ξ_i tending to 0 which enforces all constraints and all the training points are in the correct side of the margin.

This is our primal problem. If we solve the Lagrangian dual of the primal problem, we obtain the dual problem:

$$\text{Maximize } f(c_1, ..., c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j,$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda}, \forall i \in [1; n]$$

We can clearly see that the objective function is a quadratic function of the c_i subject to linear constraints, which can be efficiently solved by quadratic programming algorithms. The variables c_i are defined such that $\overrightarrow{w} = \sum_{i=1}^{n} c_i y_i \overrightarrow{x_i}$. We can also note that $c_i = 0$ if $\overrightarrow{x_i}$ lies on the right side of the hyperplane and $c_i \in]0; (2n\lambda)^{-1}[$ if $\overrightarrow{x_i}$ is on the margin boundary, it is called a support vector.

However this is possible only when the group of points are linearly separable from one another. If the group of points are not linearly separable then we have to map the set of points into higher dimension ($\geq N$) and then construct the linear margin. This feature of SVM uses kernel trick which avoids mapping the points to a higher dimension separately but the SVM convex optimisation problem can be solved by calculating the inner product in the original space and finding its counterpart in the higher dimension by a specified kernel formula. We have used the gaussian RBF kernel.

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp(-\gamma \|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2), \text{ with } \gamma > 0$$

We tested every possible combination of parameter choices (there are 121 of them) using cross validation and we picked the parameters which gave the best cross-validation accuracy. If the best cross-validation accuracy is achieved with more than a couple of parameters, we calculated the $\overrightarrow{w}$ for each of these couples and we took the parameters for which combination

$\overrightarrow{w}$ is the smallest because

it means that this specific classifier had a lower generalization error than the others.

6 Image Classification using Deep Learning

We have used Transfer Learning and trained the set of ECM images using a pretrained Convolutional Neural Net. Transfer learning is a machine learning method where a model developed

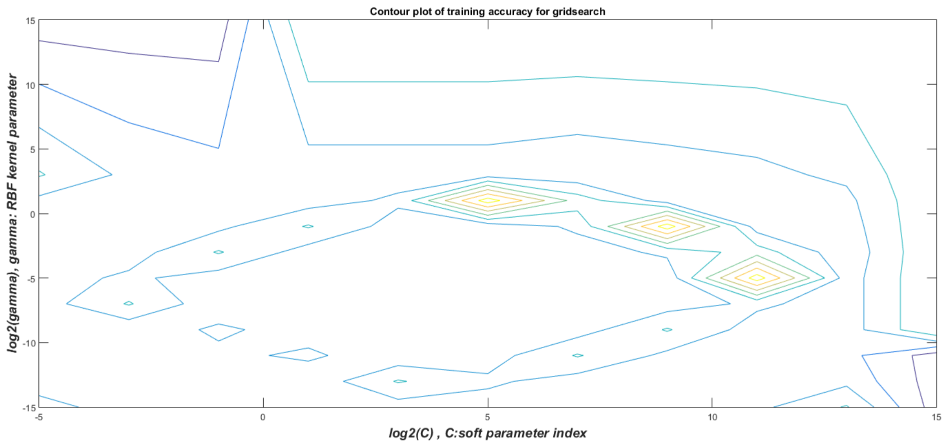


Figure 11: Contour of classfication accuracy in the grid search for optimum set of (C,γ). Here we have 3 optimum values of (C,γ) so we chose that pair which returns the lowest value of $|\overrightarrow{w}|$

for a task is reused as the starting point for a model on a second task [14] .A pre-trained model approach in Transfer Learning includes three parts-
1) Selecting the source model- A pre-trained model is to be selected which has been trained on numerous classes before. For our case we chose GoogleNet, the winner of ILSVRC (ImageNet Large Scale Visual Recognition Competition) which is developed by Google.
2) Reuse Model- The pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used. For our purpose we changed the necessary layers of the fully connected network to update the weights and bias by backpropagation in those selected layers only (refer to Section 6.1.5).
3) Tune Model- The most important phase is the tuning of the hyperparameters concerned. We chose a set of optimum hyperparameters by trail and error method to get the maximum training accuracy.

6.1 Structure of Convolutional Neural Net

A Convolutional Neural Net is made up of sequence of layers and every layer transforms an input 3D volume to an output 3D volume with some function that may or may not have parameters. We use three main types of layers to build this architecture: Convolutional Layer, Pooling Layer and Fully Connected Layer(like in regular Neural Network) [15]. We stack this sub-layers in order to form a full Convolutional Neural Net. A general Convolutional Net has the following layers-
1) Input Layer- This layer holds the raw pixel value of the images. In our case it is 224*224*3 as GoogleNet takes an RGB image of this size as it's input. So we convert our grayscale image into RGB image by adding extra two dimensions to it. These added dimensions has the same pixel intensity distribution as the original grayscale image.

2) Convolutional Layer- This layer contains filter kernels that extract specific features of the

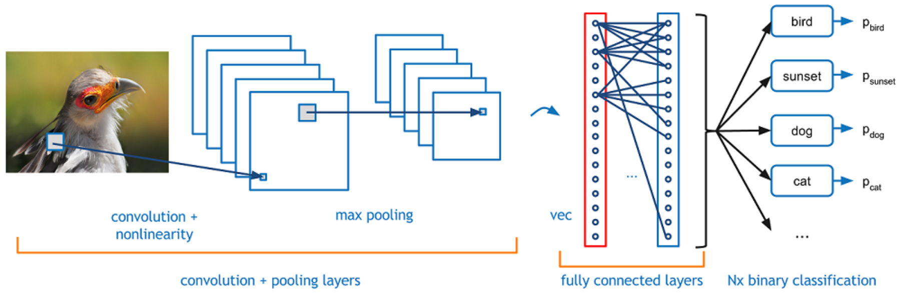


Figure 12: Architecture of Convolutional Neural Network

image. After the filters have passed over the image, a feature map is generated for each filter.
3)Non-linearity Layer(RELU)- It applies an elementwise activation function such as the max(0,x) thresholding function. All the negative pixel values are forced to 0.
4)Pooling Layer- This Pool layer performs a downsampling operation along the spatial dimensions resulting in decrease in volume. Pooling can be both MaxPooling or Average Pooling.
5) Fully Connected Layer- This is a Fully Connected Standard Neural Net for classification. The Fully Coonected Layer consist of:
a) Input layer: The last feature map of the convolutional layer forms the input layer
b) Hidden layers: There are numerous hidden layers that are fully connected with weights (for details refer to Section 6.1.5)
c)Output Layer: The last layer contains nodes corresponding to each of the variants of the image. The nodes in this layer shows the probablity that the classifier can tell it belongs to a particular class.
The last layer of the Fully Connected Layer consists of four nodes each corresponding to an ECM variant.

6.1.1 Input layer

The input layer contains a 3 channeled input or an RGB image of size (224*224*3). In our case all the images in our database (280) consisting of both training and test images are fed to Convolutional Neural Net. In our experiment all the scanner and confocal images are grayscale image. To convert them into RGB, we introduce two extra channels that have the same pixel intensity distribution as the orignal grayscale one. However for training the classifier and updating the weights, only the training set is fed. The whole database is splitted in 7:3 ratio and separated into training (196) and test set (84). We have performed the whole process of classification with different sets of test images and training images. That means the images that are in test set in the first round is not included in the test set in the second round. Like this all the images in our dataset has been only once in the test set. Like this we eradicate any bias (this 'bias' is different from the other parameter also known as 'bias' that is being updated in backpropagation) that has been shown to any particular set of images.

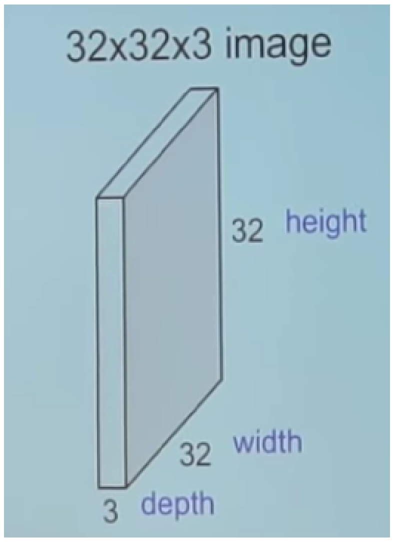


Figure 13: Example of input image
In our experiment it is 224*224*3 [6]

6.1.2 Convolution

In the visual demonstration, suppose we take a 5*3*3 filter and slide it over the image and along the way take the dot product between the filter and chunks of the input image.

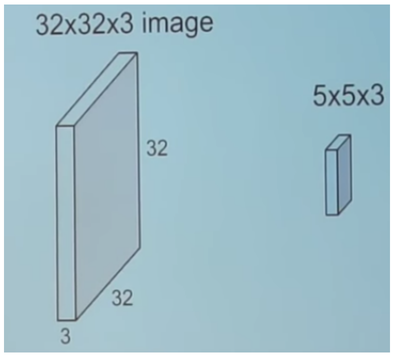


Figure 14: Convolving an image with a single filter [6]

The first part of convolution is taking the dot product with the chunk of the input image with the filter. This results in a single number as shown in the figure as shown in Fig-13.

Then the single filter is slided over all the portions of the input image such that it covers the whole image and on it's way it goes along taking the dot product. This combined process of taking the dot product and sliding through the whole of the input image is called image convolution. The output image of the convolution is known as the 'Feature Map' of the input image. It is to be noted that the size of the feature map is different from the size of the input image. This

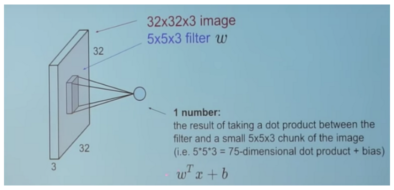

32x32x3 image
5x5x3 filter w
32
32
3
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T x + b$

Figure 15: Taking dot product with s single filter and a chunk of the input image [6]

new size is dependent on the size of both input image and the filter. In our demonstration if we have input image of size 32*32*3 and filter of size 5*5*3, then the generated feature map is of size 28*28*1.

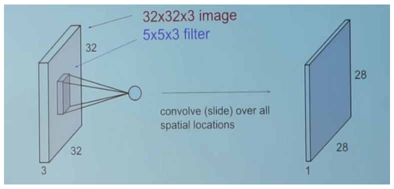


Figure 16: Sliding the filter over the input image and taking the dot product results in a new image called the Feature Map.[6]

Each feature map demonstrates a unique feature of the input image like edge, orientation towards a specific direction etc. This feature is extracted through the filter. So to extract various features we apply various filter to the input image and hence obtain numerous feature maps. If we apply 6 filters in the first convolutional layer, the resultant image dimension is 28*28*6. In the next convolutional layer, we apply a filter of dimension of 5*5*6 which results in a feature map of dimension 24*24*1. Using 10 such 5*5*6 filters will give us an output dimension of 24*24*10. It is to noted that the dot product occurs between the feature map of dimension 28*28*6 and the filter of dimension 5*5*6 to result in a feature map of dimension 24*24*1. So the depth of the volume of image depends only on the number of filters we use.

6.1.3 Non linearity layer

Just after the feature map is generated a ReLu function is applied on the image. Generally the function is of the form max(0,x). It leaves the positive pixel values of the feature map unchanged but the negative pixel values are forced to zero. Now if we have convolutional layers in sequence followed by ReLU , the size of feature maps decrease as we go deep into the network.

In CNN's we use multiple layers of convolutional layers, in each layer a specific set of filters are applied. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers.

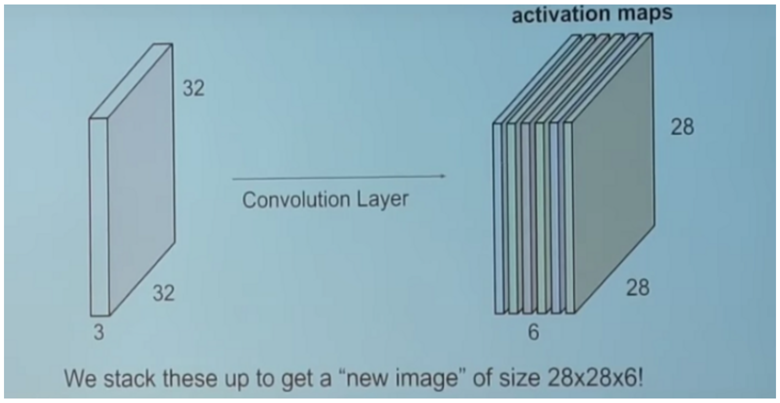


Figure 17: In this demonstration if we use 6 filters on the input image we get a new image of size 28*28*6. [6]

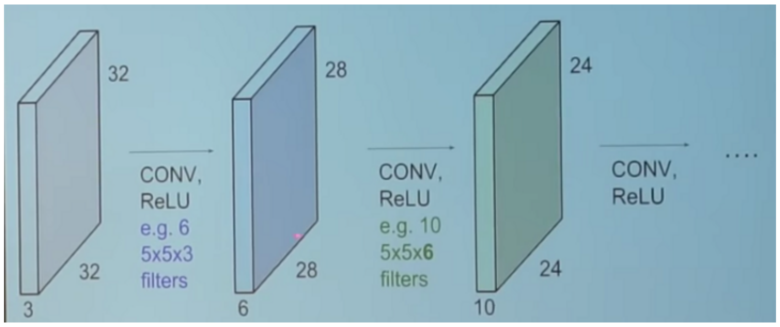


Figure 18: Adding more and more convolutional layers followed by ReLU decreases the size of the feature map [6]

6.1.4 Pooling layers

The pooling layer is used to reduce the spatial size of the feature maps after the convolutional layers. This pooling layer is used to reduce the amount of parameters and hence computations when we finally proceed to the fully connected neural network part.

Max Pooling is the most common approach and GoogleNet uses Max Pooling.

6.1.5 Fully Connected Layer

The feature map in the last layer of the convolutional neural net is of much reduced size than the original image because of subsequent convolutional and pooling operation. All the pixel values in the last layer of feature map is streched (from 3d volume to a single array consisting of nodes) to form the input nodes of a fully connected neural network. The Fully Connected Neural Network has several hidden layers and an output layer consisting of the number of variants.

Each node in a particular layer is connected by all the nodes in the previous layer by 'weights' and a separate term called' bias' is added to that node which is a term independent of the previous

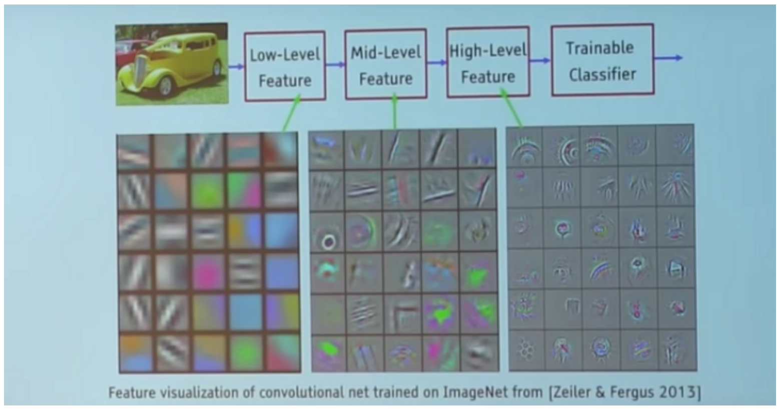

Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Figure 19: Various filters are used in each layer of the CNN [6]

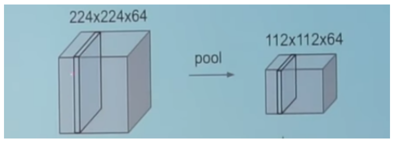


Figure 20: Pooling reduces the spatial dimension of the feature map but the depth remains unchanged [6]

node. By nodes here we refer to a single value. It can also be said that value of particular node in a layer is a linear combination of the values of the nodes in its previous layer plus an offset term known as bias. Here $Z_{l,j+1}$ is the lth node in the $(j+1)th$ layer, which is a linear combination of all the nodes in the jth layer and is represented by $Z_{i,j}$ where i=1,2...N (N is number of nodes in layer j) and plus an offset term $'b'$. G is the non-linear acticvation function (ReLU).

$$Z_{l,j+1} = G(\sum_{i=1}^{N} W_i Z_{i,j} + b)$$

(16)

Calculating the value of the nodes in the next layer from the values of the nodes in the previous layer is known as 'Forward Propagation'. It is a process of feeding input values to the neural network and getting an output which we call predicted value. Second layer takes values from first layer and applies multiplication, addition and activation operations (applying non-linear functions) and passes this value to the next layer. Same process repeats for subsequent layers and finally we get an output value from the last layer. [5]

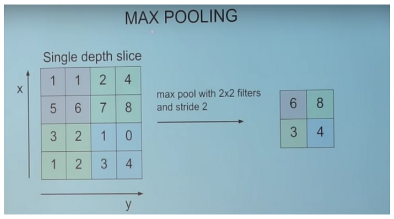


Figure 21: Example of Max Pooling [6]

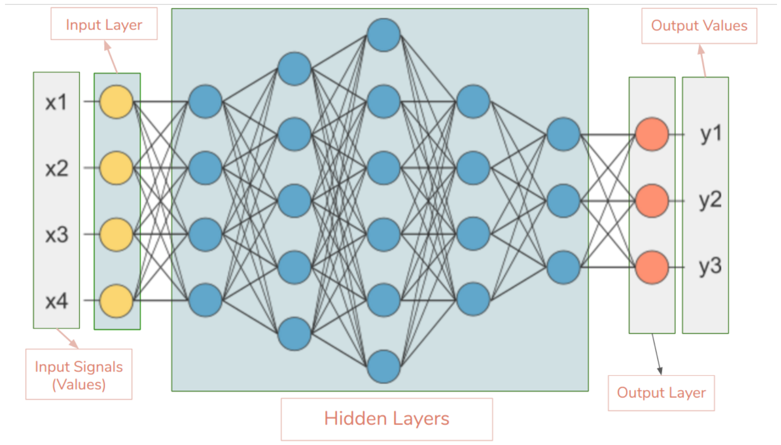


Figure 22: In a Fully Connected Neural Network, each node in the hidden layer is connected to all the nodes in the previous layer [5]

After forward propagation we get an output value which is the predicted value. To calculate error we compare the predicted value with the actual output value. We use a loss function (mentioned below) to calculate the error value. Then we calculate the derivative of the error value with respect to each and every weight in the neural network. Back-Propagation uses chain rule of Differential Calculus. In chain rule first we calculate the derivatives of error value with respect to the weight values of the last layer. We call these derivatives, gradients and use these gradient values to calculate the gradients of the second last layer. We repeat this process until we get gradients for each and every weight in our neural network. Then we subtract this gradient value from the weight value to reduce the error value. In this way we move closer (descent) to the Local Minima(means minimum loss) [5]. Suppose the output of the fully connected layer are p1, p2,p3 and p4. It is mentioned later that the loss function is a function of all 4 variables p1,p2,p3 and p4 (21). The probablities of the four nodes are obtained by passing the linear combination of the previous node values $Z_{k,1}$ thorugh an Activation function G (ReLU function).

$$p_i = G(\sum_{k=1}^{N} W_{k,1,i} Z_{k,1} + b_i)$$

(17)

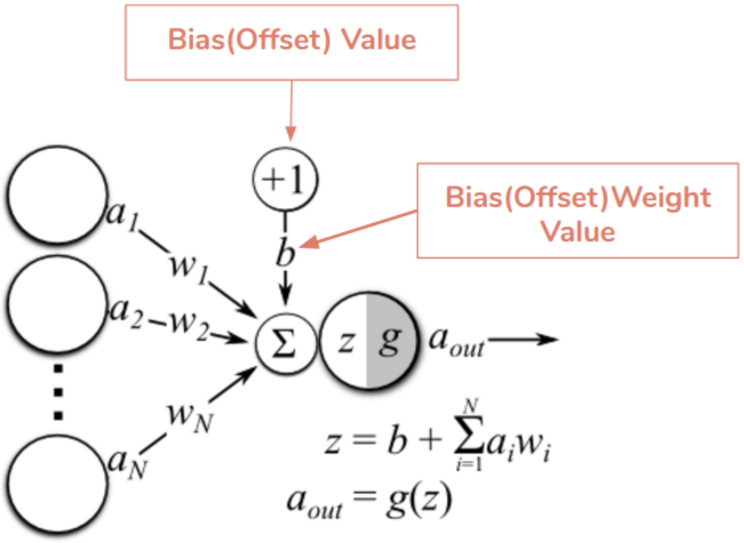


Figure 23: The input node value are represented as a_i which are multiplied by weights w_i added to form a linear combination of a_i's , a separate bias term b is added to the resulting value and then the final value is passed thorugh an activation function to introduce non-linearity (sigmoid/tanh/ReLU)[5]

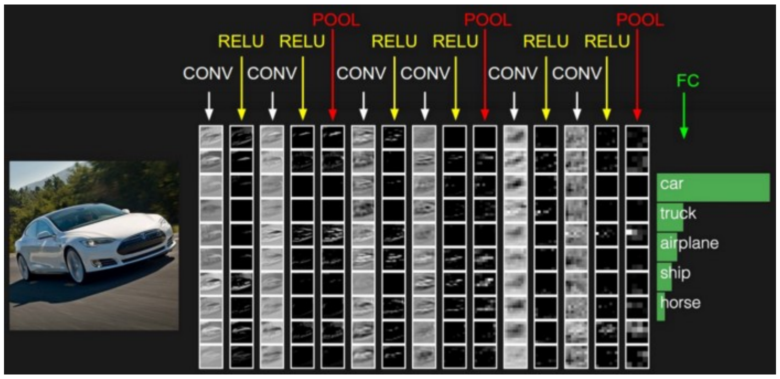


Figure 24: Stacking all the layers described above to form the whole Convolutional Neural Net

$W_{k,1,i}$ determines the weight attached to the kth node of the 1st hidden layer just preceding the output layer and is connected to the ith output node. $Z_{k,1}$ is value of the kth node in the 1st hidden layer just preceding the output layers. The 1st hidden layer preceeding the output layer has N nodes. Similarly, equation 18 describes how the value of the nodes in the first hidden layer is dependent on the values of the nodes in the second hidden layer preceeding the output layer. The second hidden layer preceeding the output layer has M nodes.

$$Z_{k,1} = G(\sum_{l=1}^{M} W_{l,2,k} Z_{l,2} + b_k)$$

(18)

Our goal with backpropagation is to update each of the weights in the network so that they cause the predicted output (p_{ij})to be closer to the actual output (y_{ij}), thereby minimizing the Loss function. To know how much change in $W_{k,1,i}$ affects the loss function w calculate the partial derivative $\frac{\partial Loss}{\partial W_{k,1,i}}$. We do this by applying the chain rule of differentiation.

$$\frac{\partial Loss}{\partial W_{k,1,i}} = \frac{\partial Loss}{\partial p_i} . \frac{\partial p_i}{\partial W_{k,1,i}} \text{for k=1,2...N and i=1,2,3,4}$$

(19)

With the help of 19, we get the derivative of all the weights that connects the first hidden layer to the final output layer.

Gradient descent is an iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. To update the weights in the direction of the minimum loss function, we use 20 where η is the learning rate which is discussed in Section 6.3.1.

$$\mathrm{W}_{k,1,i}(updated) = W_{k,1,i} - \eta \frac{\partial Loss}{\partial W_{k,1,i}}$$

(20)

This set of updated weights are then used for forward feedback for the next epoch. Like this more the number of epochs we use we keep on updating the weights in a direction that minimises the loss function and give better classification accuracy. Here, we have only shown how the weights between the first hidden layer and the output layers are updated, similarly the weights of the previous layers can also be updated through the same method of long chain differentiation.

6.2 Architecture of GoogleNet

GoogleNet or the Inception Model is the winner of ILSVRC-2014. It achieved a top-5 error rate of 6.67% ! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was actually rather hard to do and required some human training in order to beat GoogLeNets accuracy. The key innovation on the inception models is called the inception module. This is a block of parallel convolutional layers with different sized filters (e.g. 11, 33, 55) and a 33 max pooling layer, the results of which are then concatenated.The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million [10].

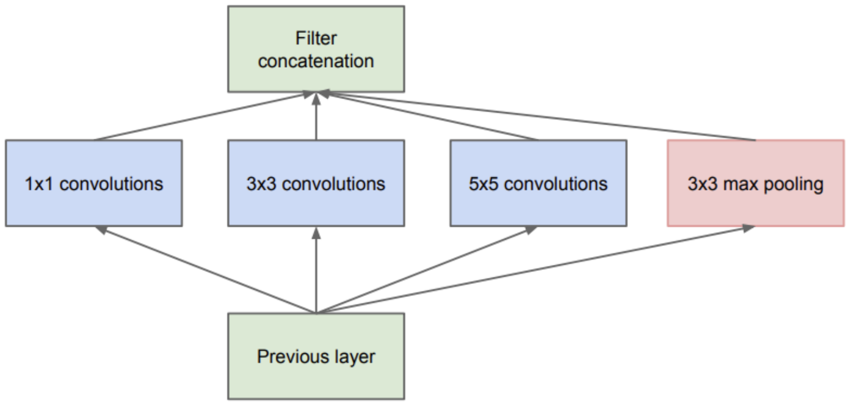


Figure 25: Example of Inception Module

6.3 Tuning the hyperparameter of the models

Deep learning models are full of hyper-parameters and finding the best configuration for these parameters in such a high dimensional space is not a trivial challenge.These hyper-parameters act as knobs which can be tweaked during the training of the model. For our model to provide best result, we need to find the optimal value of these hyper-parameters. Our main motive is to update the weights in the layers of the Fully Conncted Layer. Loss function in Multi-Class

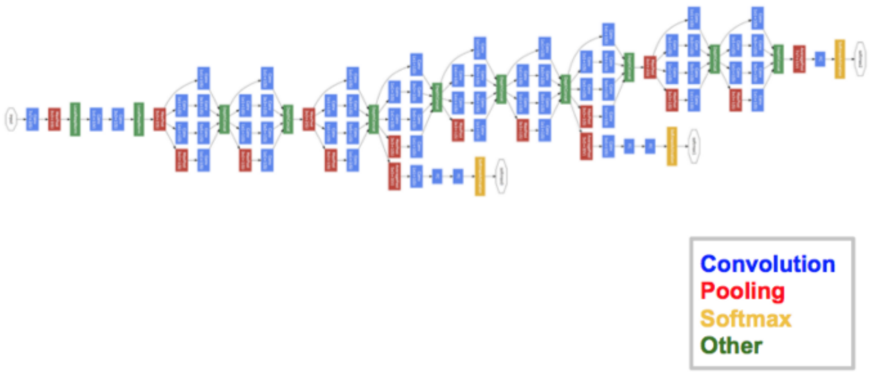


Figure 26: GoogleNet Architecture

Classification problem is called the Cross Entropy Loss or the Log-loss.

$$Loss = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log p_{ij} \tag{21}$$

Suppose there are M output classes (here 4) and we test N images (here 84). We also assign labels to the variants for example for variants A+,A-B-,B+ and A+B+ the values of j are 1,2,3 and 4 respectively. In testing of the ith image , the jth output node shows the probablity p_{ij}. This is the probablity that image i belongs to the jth class where i=1,2,3,..84 and j=1,2,3,4. Let y_{ij} be actual binary indicator that the ith test image belongs to jth class or not. For example if the 20th test image actually belongs to class A-B-, then $y_{20,1} = 0, y_{20,2} = 1, y_{20,3} = 0$ and $y_{20,4} = 0$. With the above notations, the loss function is a good measure of how the prediction done by the classifier deviates from the original label. Say for the 20th image mentioned above the $p_{20,j}$ values are 0.6, 0.1, 0.2,0.1 respectively, that means it shows a probablity 0.6 for variant A+, 0.1 for variant A-B-, 0.2 for variant B+ and 0.1 for variant A+B+. Then the loss value is $log(0.1)$ or 10 which is high. This shows that the probablity estimates for the classes are poor because of the high loss value. However if the $p_{20,j}$ values would have been 0,0.9,0.1 and 0 for j=1,2,3 and 4 respectively. Then the loss value would be 0.045 which is very low and is an indication that the predicted value(0.9) and the true value (1) for output node of A-B- are close.

. The three important parameters that has been tuned in our experiment to achieve best classification accuracy are-

6.3.1 Learning rate

In the gradient descent algorithm, we start with random model parameters and calculate the error for each learning iteration, keep updating the model parameters to move closer to the values that results in minimum cost. Gradient descent algorithms multiply the gradient (slope) by a scalar known as the learning rate (or step size) to determine the next point. This parameter tells how far to move the weights in the direction of the gradient. In other words learning rate

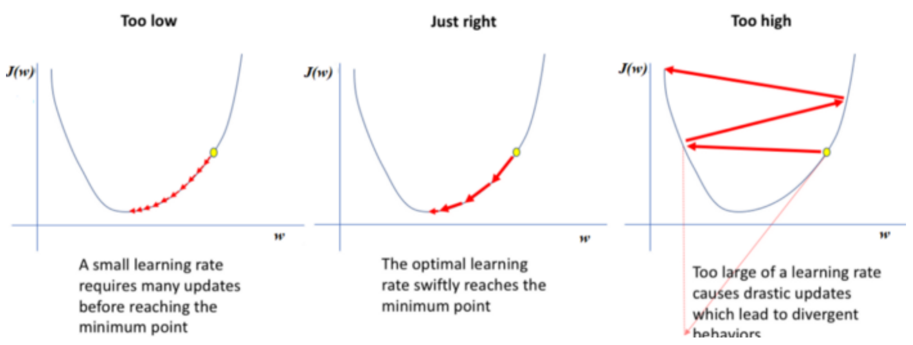

Too low
Just right
Too high
J(w)
J(w)
J(w)
w
w
w
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure 27: Effect of various values of learning rate [7]

controls the rate or speed at which the model learns. If the learning rate is small, then training is more reliable, but it will take a lot of time because steps towards the minimum of the loss function are tiny. If the learning rate is high, then training may not converge or even diverge. Weight changes can be so big that the optimizer overshoots the minimum and makes the loss worse. At extremes, a learning rate that is too large will result in weight updates that will be too large and the performance of the model (such as its loss on the training dataset) will oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge (are divergent)[7]. Thus our aim is to find the optimal learning rate which can quickly find the minimum loss.

6.3.2 Number of Epoch

One Epoch is when an ENTIRE dataset is passed through the neural network only ONCE. Updating the weights with a single pass or one epoch is not enough and may lead to underfitting. As the number of epochs increases, more number of times the weight are changed in the neural network and the curve goes from underfitting to optimal to overfitting curve[7]. So by fine-tuning it is important to set a specific value of epoch to get an optimum classification accuracy.

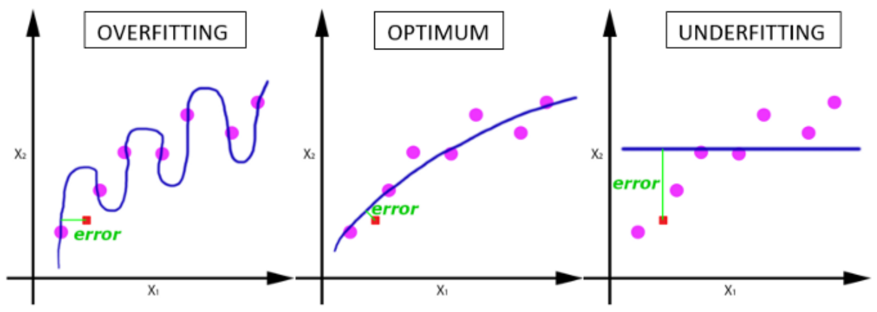


Figure 28: Effect of number of epochs[7]

6.3.3 Batch-size

One can't pass the entire dataset into the classifier because of limited memory. So we divide the entire dataset into batches. When each batch of training set is fed to the classifier it is called one iteration. After several batches (which makes up the entire dataset) have been fed to the classifier it is called one epoch. Thus one epoch is composed of several iterations.

7 Classification results

We have applied both the classification methods using curvelet feature extraction and deep learning on four sets of data-
1) Images acquired by Confocal Microscope (referred to as the 'Old Confocal Set')
2) Images acquired by Confocal Micrscope and treated with antibody-A (referred to as 'Set-12A')
3) Imgaes acquired by Confocal Microscope and treated with antibody-B (referred to as 'Set-12B')
4) Images acquired by Coverslip Scanner (referred to as 'Scanner Images')

It is to be noted that the results demonstrated here are the optimum one i.e for a particular set using a specific classification method, the parameters are the optimum one and the classification accuracy is the maximum for this optimum parameter among any other combination of parameters.
In Curvelet based classification method, the whole database containing 280 images had been split into 224 images into the training set and 56 images into the test set. We performed a 5-fold cross validation which means we performed the classification 5 times and in each of those 5 times we used a separate test set for classification. So each image had been once in our test set. This is done to remove the bias towards a particular set of images for classification. The final classification accuracy is the mean of the 5 individual classification accuracies.Size of the images in curvelet based feature extraction is 512*512.

In deep learning based classification method, the whole database containing 280 images has been split 196 images into the training set and 84 images into the test set. Here we performed a 3-fold cross validation which means we performed the classification 3 times and in each of those three experiments, we have used separate test set of images. In each experiment (out of 3), the number of epochs have been set. In each epoch, the whole of the training set (196) is used for classfication. However as this size is large, we divide 196 training sets into batches. Passing of each batch of training images into classifier is called one iteration. To sum it up, each experiment consists of a number of epochs and each epoch consists number of iterations. The final classification accuracy is the mean of the 3 individual classification accuracies. Size of the images in deep learning based feature extraction is 224*224*3.

In the following subsections, classification results are shown for each set using both curvelet based feature extraction and deep learning.

7.1 Results for Old Confocal Microscope Set

Table 1: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
(without invariance to rotation)

Actual/Predicted	A+	A-B-	B+	A+B+
A+	0.9714	0	0	0.0286
A-B-	0	0.9286	0.0429	0.0286
B+	0.0714	0.0571	0.3857	0.4857
A+B+	0.0714	0.0571	0.3857	0.4857

Mean classification accuracy= 69.28%

Table 2: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
(with invariance to rotation)

Actual/Predicted	A+	A-B-	B+	A+B+
A+	0.9286	0	0.0429	0.0286
A-B-	0	0.7857	0	0.2143
B+	0.0714	0.1571	0.2143	0.5571
A+B+	0.0571	0.1714	0.2143	0.5571

Mean classification accuracy= 62.1429%

Table 3: Classification using CNN with parameters
Batch-size= 50
Number of Epochs=20
Learning rate=0.0009

Actual/Predicted	A+	A-B-	B+	A+B+
A+	0.952	0	0	0.0476
A-B-	0	0.904	0.0909	0
B+	0	0.095	0.6190	0.2857
A+B+	0	0.1428	0.3857	0.4761

Mean classification accuracy= 73.57% (best among the three)

7.2 Results for Confocal Micrscope Set-12A

Table 4: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
(without invariance to rotation)

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.500	0	0.2571	0.2429
A-B-	0	0.9143	0.0857	0
B+	0.2857	0.1571	0.3143	0.2429
A+B+	0.1571	0.0429	0.3143	0.4857

Mean classification accuracy= 55.357%

Table 5: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
(with invariance to rotation)

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.4714	0.0571	0.1714	0.3000
A-B-	0.0286	0.9429	0.0286	0
B+	0.2714	0.2143	0.2857	0.2286
A+B+	0.2286	0.0714	0.2429	0.4571

Mean classification accuracy= 53.9286%

Table 6: Classification using CNN with parameters
Batch-size= 50
Number of Epochs=20
Learning rate=0.0009

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.904	0	0.0476	0.0476
A-B-	0	0.952	0	0.0476
B+	0.0476	0	0.7619	0.1428
A+B+	0.238	0	0.238	0.5238

Mean classification accuracy= 78.57 % (best among the three)

7.3 Results for Confocal Micrscope Set-12B

Table 7: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.75
(without invariance to rotation)

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.4286	0.2143	0.22286	0.1286
A-B-	0.1429	0.7571	0.0571	0.0429
B+	0.1429	0.1714	0.4286	0.2571
A+B+	0.185	0.1143	0.2857	0.4143

Mean classification accuracy= 50.7143%

Table 8: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.75
(with invariance to rotation)

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.4714	0.2286	0.2143	0.0857
A-B-	0.2429	0.3286	0.1714	0.2571
B+	0.1286	0.0571	0.800	0.0143
A+B+	0.1714	0.2714	0.1857	0.3714

Mean classification accuracy= 49.2857%

Table 9: Classification using CNN with parameters
Batch-size= 50
Number of Epochs=20
Learning rate=0.0019

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.7673	0.2857	0	0
A-B-	0	1	0	0
B+	0.333	0.190	0.238	0.190
A+B+	0.333	0.238	0.238	0.238

Mean classification accuracy= 58.33% best among the three

7.4 Results for Coverslip Scanner

Table 10: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.7286	0	0	0.2714
A-B-	0	0.8571	0.0857	0.0571
B+	0	0.100	0.6571	0.2429
A+B+	0.1571	0	0.1143	0.7286

Mean classification accuracy= 74.2857%

Table 11: Classification using CNN with parameters
Batch-size= 50
Number of Epochs=20
Learning rate=0.0009

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.95238	0	0	0.0476
A-B-	0	1	0	0
B+	0	0.047	0.6190	0.333
A+B+	0	0	0	1

Mean classification accuracy= 89.29% (better)

8 Comparision among various classification methods and acquisition methods

From the above results it is conclusive that images that were acquired by scanner were better classified by both the classifiers than the ones acquired by confocal microscope. After a close observation by biology experts, we reached to a conclusion that due to presence of more noise in the images acquired by confocal micrscope, the classfication results for the images acquired by conofocal microscope were not up to the mark like the ones acquired by coverslip scanner. So we tried to apply non-linear filters to remove the inter-region noise but keeping the curves and edges intact. Linear filters like Low-pass filters and Gaussian filters are avoided because then the edges and the curves of the ECM would get blurred. It has been found that this type of denoising problem have been well addressed by anisotropic diffusion filtering.

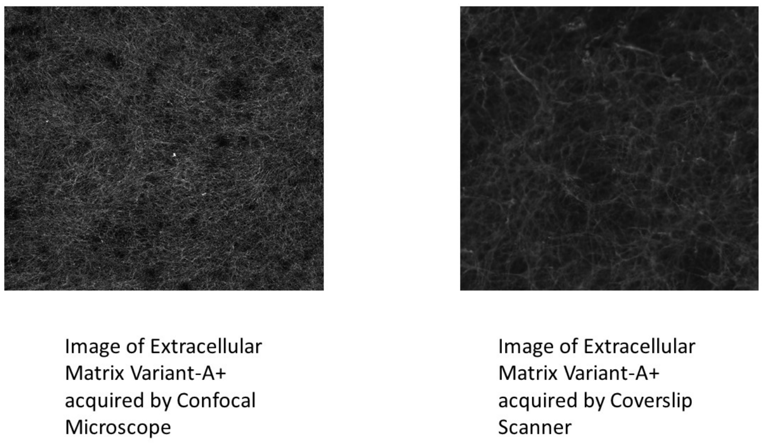


Image of Extracellular Matrix Variant-A+ acquired by Confocal Microscope
Image of Extracellular Matrix Variant-A+ acquired by Coverslip Scanner

Figure 29: Presence of noise in images acquired by confocal micrscope influences the classfication accuracy

8.1 Anisotropic diffusion filtering

In image processing and computer vision, anisotropic diffusion, also called Perona–Malik diffusion, is a technique aiming at reducing image noise without removing significant parts of the image content, typically edges, lines or other details that are important for the interpretation of the image [11]. Kappa or the conduction coefficient controls conduction as a function of gradient. If kappa is low small intensity gradients are able to block conduction and hence blocking diffusion across steep edges. Whereas a large value of kappa allows conduction across the steep edges, So kappa is set at an optimum value such that we can denoise the inter-region portion without significantly effecting the edges.

We have run the classification test for various values of kappa. The classification result for the optimum kappa is shown here:-

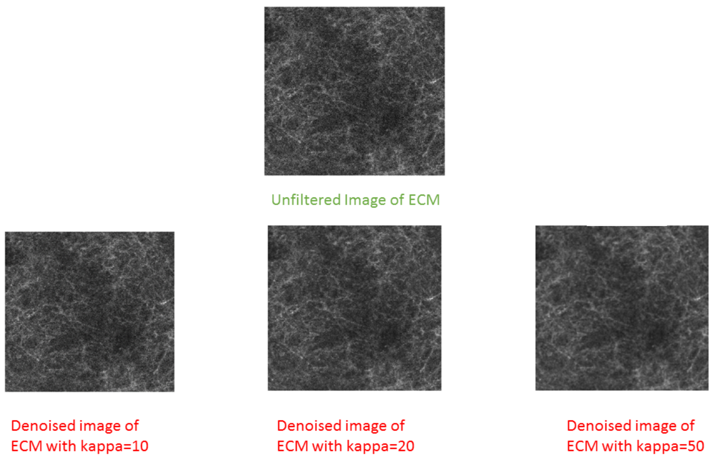

Unfiltered Image of ECM
Denoised image of ECM with kappa=10
Denoised image of ECM with kappa=20
Denoised image of ECM with kappa=50

Figure 30: Effect of ECM image on various values of conduction coefficient kappa

8.2 Classification of Old Confocal Images with and without filtering

Table 12: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
without using any filters

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.9714	0	0	0.0286
A-B-	0	0.9286	0.0429	0.0286
B+	0.0714	0.0571	0.3857	0.4857
A+B+	0.0714	0.0571	0.3857	0.4857

Mean classification accuracy= 69.28%

Table 13: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
on pre-filtered images with
conduction coefficient (kappa) = 10

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.9857	0	0	0.0143
A-B-	0	0.9429	0	0.0143
B+	0.0429	0.0286	0.5000	0.4286
A+B+	0.0429	0.0286	0.5000	0.4286

Mean classification accuracy= 71.4286%

8.3 Classification of Set-12A images with and without filtering

Table 14: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
without using filter

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.500	0	0.2571	0.2429
A-B-	0	0.9143	0.0857	0
B+	0.2857	0.1571	0.3143	0.2429
A+B+	0.1571	0.0429	0.3143	0.4857

Mean classification accuracy= 55.357%

Table 15: Classification using Curvelet based feature extraction with parameters
k=400 and
pctg=0.85
on pre-filtered images with
conduction coefficient (kappa) = 10

Actual/Predicted	**A+**	**A-B-**	**B+**	**A+B+**
A+	0.8571	0	0.0714	0.0714
A-B-	0	0.928	0.0714	0
B+	0.0714	0	0.571	0.3571
A+B+	0	0.0714	0.4285	0.5

Mean classification accuracy= 71.23%

8.4 Apparent confusion between variants B+ and A+B+

It is evident that the images acquired by coverslip scanner showed improved classification results by both curvelet based feature extraction and deep learning. To address this problem we even applied anisotropic diffusion filtering which improved the classfication results of images acquired by confocal micrscope but not to the extent which has been shown by images acquired by coverslip scanner.
However in all of the tests, there is always an apparent confusion made by the classifier in distinguishing variants B+ and A+B+. The reason being both variants has similar topological structure to some extent. So we decided to perform deep-learning based classification on images acquired by coverslip scanner using only 3 variants. The first two variants are A+ and A-B-. The third variant can be among the following three-
1) Only containing images from variant B+
2) Only containing images from variant A+B+
3) A hybrid set containing images from both the variants B+ and A+B+

8.5 Classification results using three variants for images acquired by coverslip scanner

Table 16: Classification using deep-learning
Third variant= B+

Actual/Predicted	A+	A-B-	B+
A+	1	0	0
A-B-	0	0.952	0.048
B+	0	0	1

Mean classification accuracy= 96.83%

Table 17: Classification using deep-learning
Third variant= A+B+

Actual/Predicted	A+	A-B-	A+B+
A+	0.9047	0	0.0953
A-B-	0	1	0
A+B+	0	0	1

Mean classification accuracy= 93.65%

Table 18: Classification using deep-learning
Third variant= mixed class of B+ and A+B+ (mix)

Actual/Predicted	A+	A-B-	mix
A+	1	0	0
A-B-	0	1	0
mix	0	0	1

Mean classification accuracy= 100%

9 Conclusion for the Results

In this section we draw a detailed scientific conclusion on all the results we have achieved. It also includes the reasons why some classfication methods or acquisition methods performed better than others and the sensitivity of the parameters in the classification.

9.1 Images acquired by confocal microscope vs images acquired by coverslip scanner

Comaparing section 7.4 with 7.3,7.2 and 7.1, it is evident that images acquired with coverslip scanner were easier to classify than the ones acquired with confocal microscope.
In the confocal micrscope acquisition, the images are taken in slots and that slot is moved from region to another region of the cell underneath to cover the whole of the cell. In each slot, the

whole depth of the cell is taken into account, i.e the image is an integration of all the fibres present throughout that depth in that particular slot.
In the coverslip scanner acquisition, the whole image of the cell is taken with a depth adjusted to the focus of the scanner. Thus some region of the cell is out of focus and we had to neglect those part for our experiment.
In images acquired by confocal micrscope, there is presence of more inter-region noise because it integrates throughout the depth of the slot of the cell.
This maybe a possible reason why we have better classification results using coverslip scanner than with confocal microscope.
Same set of Extracellular matrix were acquired with confocal micrscope (labelled as 'Set-12') and with coverslip scanner (labelled just as scanner). Set-12 had two sets of experiments namely-Set-12A and Set-12B , each being treated with a separate antibody and hence are a part of two different experiments.

9.2 Classification methods: Curvelet based feature extraction vs Deep Learning

For all the three sets of experiments, deep learning has outperformed curvelet based classification methods by a significant margin of mean classification accuracy. We have used a pre-trained Googlenet to classify the four variants of ECM images. Deep learning techniques learn by creating a more abstract representation of data as the network grows deeper, as a result the model automatically extracts features and yields higher accuracy results. Convolutional neural net loosely mimics the brain function ,multiple layers of neural networks stacked one after operates as a classical brain model. GooglNet achieved a top-5 error rate of 6.67%. It required some human training in order to beat GoogLeNets accuracy. Even in our experiment it could classify the variants A+ and A-B- with a very good accuracy and only has problem distinguishing B+ and A+B+. This leads us to a conclusion that topological structures of the variants B+ and A+B+ are very similar in nature and they behave like a single variant. However curvelets do a fine job in representing the scale and orientations of the fibre as a feature vector but classifying a large amount of feature vectors using DAG-SVM does not result in higher accuracy like deep-learning. Although deep learning has a greater accuracy as compared to curvelet based extraction, however it fails to explain the features of each variant that are responsible for distinguishing them which have been done nicely by the curvelets.

9.3 Anisotropic Diffusion filtering: Unfiltered Confocal Set vs Pre-filtered Confocal Set(only for curvelet based feature extraction)

After observing that there was a significant difference in the classification accuracy of confocal and scanner set, it was decided that this difference was due to presence of inter-region noise in the images acquired by the confocal set. So to filter out the noise effect, we performed a non-filtering operation known as anisotropic diffusion filtering. These types of filters are used when we have to denoise inter-region part of the images without affecting the edges, curves of the images. With the help of anisotropic diffusion filtering, the classification accuracy for

Table 19: Sensitivity of k to accuracy and execution time (pctg=85%)

k (Number of Centroids)	Mean Accuracy	Execution Time (mins)
400	69.28%	34
450	69.64%	37
500	69.72%	42

Old Confocal Images increased by 2.14% and that of the Set-12A confocal images increased by 15.83%. The improved classification accuracy for the confocal set (71%) is quite close to that of the scanner set (74%).

9.4 Sensitivity of parameters to classification accuracy

As all the calculations are performed by the software, it is our prime task to set the paramters of classification to an optimum value. This is done by trail and error method , however some parameters are more sensitive than the rest and not setting that parameter to an optimum can result in significant decrease of classification accuracy.

9.4.1 Sensitivity of parameters in Curvelet based feature extraction

Pctg or percentage of curvelet coefficients kept is the most important parameter in curvelet based feature extraction. Increasing pctg means we have more curvelet coefficients which increases the classification accuracy as we take a better approximate of the image into account. But it comes at the expense of increased computational time. Moreover we have limited memory for storage and hence increasing pctg above 95% would result in memory overflow.
Number of centroids is the second most important parameter and dictates the size of the image signature. Increasing number of centroids results in decrease of SSE (Sum of Squared Error) and makes a good approximation of the keypoint curvelet features. But it comes at the expense of increased computational time. Moreover we would also be adding more dimensions in the SVM problem. Inclusion of poor features may cause SVM to learn the noise in the data, damaging the accuracy.
The combination of parameters pctg=85% and k=400 showed maximum classification accuracy for Set-12 A Confocal Set and Scanner images. However this set of parameters resulted in memory overflow for the Set-12B images. So we took a revised set of paramters pctg=75% and k=400 for our experiment with Set-12B. (This may be a reason why classfication results for Set-12B is the least.)
In case of Old Confocal Image Classification with curvelets, increasing k from 400 to 450 and keeping other parameters constant, increased classfication accuracy from 69.28% to 69.64% , but the execution time of the classification also increased from 34 minutes to 37 minutes. Again keeping k=400 constant and varying pctg from 80% to 85% increased classification accuracy from 66% to 69% but execution time increases from from 30 mins to 34 mins.

Table 20: Sensitivity of Learning Rate in Deep Learning based classification for Scanner images (Batch size= 50, Number of epochs= 20)

Learning rate	**Classification Accuracy**
0.0003	82.33%
0.0009	89.29%
0.0019	72.46%

Table 21: Sensitivity of Number of Epochs in Deep Learning based classification for Scanner images (Batch size= 50, Learning rate=0.0009)

Number of Epochs	**Classification Accuracy**
10	75.68%
20	89.29%
25	84.23%

9.4.2 Sensitivity of parameters in Deep Learning

Learning rate is the most important and sensitive tunable parameter in deep learning based classification. The influence of learning rate on classification is described in Section 6.3.1. In Table 20, it is shown that classification accuracy reaches a maximum at learning rate=0.0009 and increasing the rate above this point would result in a significant decrease in classification accuracy. However the optimum learning rate varies from one training set to another training set. For example for Scanner and Set-12A, the optimum learning rate is 0.0009 whereas for Set12B , it is 0.0019.

Number of epochs is the second most important parameter and the optimum number of epoch is found out to be 20 for the scanner images. However keeping the other two parameters constant, decreasing the number of epoch results leads to decrease in classification results due to underfitting. Again increasing the number of epochs, would result in overfitting as shown in Table 21.

References

[1] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," ACM Transactions on Intelligent Systems and Technology, vol. 2, pp. 27:1–27:27, 2011. Software available at
`http://www.csie.ntu.edu.tw/ cjlin/libsvm.`

[2] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms."
`http://www.vlfeat.org/,2008`

[3] E. Candès, L. Demanet, D. Donoho, and L. Ying, “Fast discrete curvelet transforms,” Multiscale Modeling and Simulation, vol. 5, pp. 861–899, 2006.

[4] Anca-Ioana Grapa, Raphael Meunier, Laure Blanc-Féraud, Georgios Efthymiou, Sébastien Schaub, et al.. Classification of the fibronectin variants with curvelets. ISBI 2018 - IEEE 15th International Symposium on Biomedical Imaging, Apr 2018, Washington, DC, United States. pp. 930-933, ff10.1109/ISBI.2018.8363723ff. ffhal-01868726

[5] Kailash Ahirwar ”Everything you need to know about Neural Networks’
`https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491`

[6] Harsh Pokhrana ”The best explanation of Convolutional Neural Networks on the Internet!”
`https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8`

[7] Javaid Nabi ”Hyper-parameter Tuning Techniques in Deep Learning”
`https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8`

[8] Siddharth Das ”CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more . . . .”
`https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5`

[9] Logloss-Wikipedia page
`http://wiki.fast.ai/index.php/LogLoss`

[10] Sik-Ho Tsang ”Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification)”
`https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7`

[11] Daniel Lopes ”Mathworks:Anisotropic Diffusion (Perona Malik)”
`https://fr.mathworks.com/matlabcentral/fileexchange/14995-anisotropic-diffusion-perona-malik`

[12] EJ Candes ”Lecture-2: Curevlets”
`http://www.cs.tut.fi/ karen/CandesCurvelets.pdf`

[13] Emmanuel Candes, Laurent Demanet, David Donoho,Lexing Ying ,”curvelet.org”
`http://www.curvelet.org/`

[14] Jason Brownie, ”A Gentle Introduction to Transfer Learning for Deep Learning”
`https://machinelearningmastery.com/transfer-learning-for-deep-learning/`

[15] CS231n Convolutional Neural Network for Visual Recognition
`http://cs231n.github.io/convolutional-networks/`